GA-based learning for rule identification in fuzzy neural networks

Dahal, Keshav; Almejalli, Khaled; Hossain, M. Alamgir; Chen, Wenbing

Published in:
Applied Soft Computing

DOI:
10.1016/j.asoc.2015.06.046

Published: 01/10/2015

Document Version
Peer reviewed version

GA-based Learning for Rule Identification in Fuzzy Neural Networks

Keshav Dahal[1], Khaled Almejalli[2], M Alamgir Hossain[3], Wenbing Chen[4]

[1]School of Engineering and Computing, University of the West of Scotland, UK

[2]Department of Computing, University of Bradford, UK

[3]IT Research Institute, Anglia Ruskin University, UK

[4]Nanjing University of Information Science and Technology, China

Email: keshav.dahal@uws.ac.uk; almejalli@gmail.com; alamgir.hossain@anglia.ac.uk; chenwb@nuist.edu.cn

Abstract— Employing an effective learning process is a critical topic in designing a fuzzy neural network, especially when expert knowledge is not available. This paper presents a Genetic Algorithm (GA) based learning approach for a specific type of fuzzy neural network. The proposed learning approach consists of three stages. In the first stage the membership functions of both input and output variables are initialized by determining their centers and widths using a self-organizing algorithm. The second stage employs the proposed GA based learning algorithm to identify the fuzzy rules while the final stage tunes the derived structure and parameters using a back-propagation learning algorithm. The capabilities of the proposed GA-based learning approach are evaluated using a well-examined benchmark example and its effectiveness is analyzed by means of a comparative study with other approaches. The usefulness of the proposed GA-based learning approach is also illustrated in a practical case study where it is used to predict the performance of road traffic control actions. Results from the benchmarking exercise and case study effectively demonstrate the ability of the proposed three stages learning approach to identify relevant fuzzy rules from a training data set with a higher prediction accuracy than alternative approaches.

Index Terms— Fuzzy rule identification, genetic algorithm, Back-propagation learning algorithm, Mamdani-type fuzzy neural network.

[1]Corresponding author: Keshav Dahal, School of Engineering and Computing, University of the West of Scotland, Paisley, PA1 2BE, UK, Email: keshav.dahal@uws.ac.uk, Tel: +44 141 848 3305.

1. INTRODUCTION

A Fuzzy Neural Network (FNN) is a hybrid intelligent system which combines the capability of fuzzy reasoning to handle uncertain information and the capability of neural networks to learn from processes [1,2]. FNN is a fuzzy system that uses the learning ability of neural networks to determine its parameters (fuzzy sets, fuzzy memberships and fuzzy rules) by processing data. An important topic in designing an FNN is that of identification of fuzzy rules [3,4,5,6]. The main aim when identifying fuzzy rules in FNNs is to learn and modify the rules from past experience. Various methods have been employed to identify the fuzzy rules in FNNs. Quek and Zhou [3] have classified rule identification methods into three categories: *i)* Methods that use linguistic information from experts [5,7]. Although this type of approach converges faster during training and performs better, it is rather subjective. *ii*) Methods that use numerical information [2,4]. These include unsupervised learning algorithms such as clustering, self-organizing and competitive learning algorithms. *iii*) Methods that use supervised learning algorithms (particularly the back-propagation technique) to identify fuzzy rules in the FNNs [6,8]. These FNNs are basically multilayered, where inputs and outputs are fuzzy membership values that satisfy certain constraints. The back-propagation learning algorithm is often utilized in such an FNN to produce the mapping from inputs to outputs. In this case the FNN appears as a black box at the end of the training process.

The use of Genetic Algorithm (GA) in the design of fuzzy systems (including rules identification, membership function adjustment, and training fuzzy rule-base models to represent specific data) has been the subject of considerably greater research effort than other approaches. This initiative has led to a fuzzy system with a learning process based on GA. GA has not only been employed to tune membership functions, but it has also been used to optimize the architecture of a FNN [9,10,11]. Wang et al. [9] proposed a GA-based approach for a feedback direct adaptive fuzzy-neural controller to tune online weighting factors. In particular, they have used a reduced-form GA to adjust the weightings of the FNN. A two-phase GA-based learning method for FNN is proposed in [10]; firstly to roughly estimate the optimal fuzzy weights, and secondly to provide better estimates for the shape of the membership function. Leng et. al [11] proposed a GA hybrid model for building a FNN system without a priori knowledge about the partitions of input space and the number of fuzzy rules. The GA attempts to identify and delete the least important neurons to yield a compact structure from the initial structure with a redundant architecture. The model also includes a hybrid learning method consisting of GA, back-propagation, and recursive least squares estimation for optimizing the initial network structure.

A learning method for fuzzy rule-based systems using the iterative rule learning approach is proposed in [12, 13]. The fuzzy rule base is constructed in an incremental fashion, where GA optimizes one fuzzy classifier rule at a time in [12], whereas the iterative genetic approach presented in [13] can include relations between variables in the antecedent of rules to improve the ability of fuzzy systems. Ishibuchi and Yamamoto [14] have proposed a GA-based approach for pattern classification problems consisting of two

phases: candidate rule generation by rule evaluation measures in data mining, and rule selection by multi-objective evolutionary algorithms.

In this paper an evolutionary GA based technique is employed within a three stage-learning approach for FNN systems, and we refer to here it as GAFNN. This proposed approach falls into the second category of rule identification methods. In the first stage of GAFNN, membership functions of both input and output variables are initialized by determining their centers and widths using a self-organizing algorithm. The GA based learning algorithm, the basic concept of which has been reported in our previous work [15], is performed in the second stage to identify the fuzzy rules. In the last stage, the derived structure and parameters are fine-tuned using the back-propagation learning algorithm.

Starting with all possible fuzzy rules then using the GA to identify the relevant rules is a very promising approach. This approach was adopted by Castro and Camagro [16] who proposed a three stage-based approach for identifying fuzzy rules from numerical data: a feature selection process, a GA for deriving fuzzy rules and, finally, another GA for optimizing the rule base. The main disadvantage of this approach is that, depending on the number of input-output variables and the number of their fuzzy sets, the total number of possible rules can be extremely large, making it difficult to encode and generate the chromosomes. Consequently, the learning process can become overloaded. However, starting the GA process with all possible rules (if possible) is still preferred because it decreases the chance of missing any relevant rule which, in turn, minimizes the final error.

The main advantage of the proposed GAFNN presented in this paper is that GA is only used to identify fuzzy rules. Using GA to optimize fuzzy membership functions and fuzzy rules simultaneously, (which is avoided in our proposed method), makes the GA suffer from the curse of dimensionality, because every fuzzy rule represents a different subspace of the input variables. Another advantage of the proposed GA-based method is that the fine tuning of the fuzzy rules weight is done in a separate learning stage (stage three). Consequently, integer representation (encoding) of the problem is used which reduces the length of the chromosome as well as making the size of the GA search space very small compared with other approaches where floating point numbers representation is implemented. The proposed GA-based learning approach is tested in a five-layer FNN with a well-examined benchmark dataset and its performance is analyzed through a comparative study with 19 other approaches reported in the literature. Furthermore, the proposed GAFNN is demonstrated in a case study which identifies appropriate road traffic control actions in the ring-road around Riyadh city. The effectiveness of the performance of GAFNN in this case study has been shown through a comparison with back-propagation NN.

2. STRUCTURE OF GAFNN

This section describes the structure and the function of the proposed GAFNN. The structure of GAFNN is of the Mamdani type and is also similar to the structure considered in [17] and [18]. The GAFNN has a total of five layers - the topology of which is shown in Fig. 1. Each layer performs an operation for building the

fuzzy system. The input and output layers are represented as vectors $X^N = [x_1, x_2, \dots, x_n, \dots, x_N]$ and $Y^P = [y_1, y_2, \dots, y_p, \dots, y_P]$, where N and P represent the number of input and output non-fuzzy variables, x_n and y_p are nth input and pth output variables respectively.

Layer 1(Input Layer): nodes at this layer are input nodes which represent input linguistic variables such as "age", "weight", and "speed" and directly transmit non-fuzzy input values to the next layer. Each node in this layer is connected to only those nodes of Layer 2, which represent the linguistic values of corresponding linguistic variables. The link weights, $W_{n,m}$ between this layer and the next layer is unity. The output $o_n^{(1)}$ of this layer is given as follows:

$$o_n^{(1)} = x_n \tag{1}$$

where x_n is the input of the input neuron L_n in Layer 1.

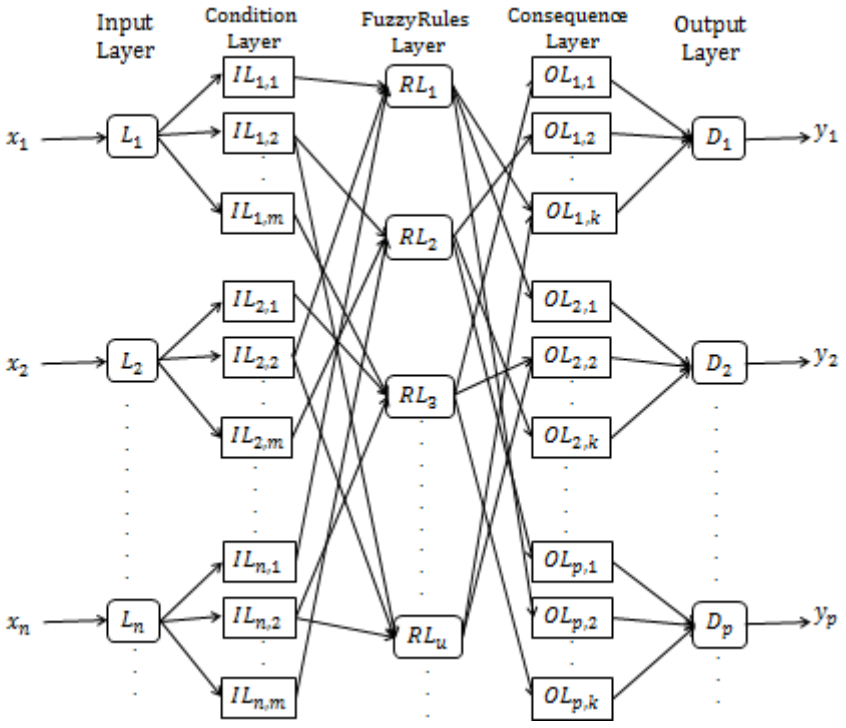


Fig. 1. Structure of GAFNN.

Layer 2 (Condition Layer): this layer defines the fuzzy sets and membership functions for each of the input factors. Nodes in this layer act as a membership function and represent the terms of the respective linguistic variable, such as "low", "medium", or "high". The input values are fed to this layer which then calculates the membership degree. In our model this is implemented using the Gaussian membership function to ensure differentiability, ensuring compatibility with the back-propagation algorithm employed in

the last stage of the learning process [18]. The connection weights in this layer are unity. The output $o_{n,m}^{(2)}$ of input-label node $IL_{n,m}$ is given as follows:

$$o_{n,m}^{(2)} = e^{-\frac{(o_n^{(1)} - c_{n,m}^{(2)})^2}{(\sigma_{n,m}^{(2)})^2}} \quad (2)$$

where $c_{n,m}^{(2)}$ and $\sigma_{n,m}^{(2)}$ are the centers (or means) and the widths (or variances) of the membership function for the input-label node $IL_{n,m}$ respectively, where $IL_{n,m}$ denotes the *m*th input label of the linguistic node *n*.

Layer 3 (Fuzzy-Rules Layer): this layer defines all of the possible fuzzy rules required to qualitatively specify how the output parameter is determined for various instances of the input parameters. Each node in this layer represents a fuzzy rule. The nodes in this layer perform the AND operation. The output $o_u^{(3)}$ of a rule node RL_u at Layer 3 is given as follows:

$$o_u^{(3)} = \min_{n,m \epsilon F} (o_{n,m}^{(2)}) \quad (3)$$

where F is the set of indices of the nodes in Layer 2 that are connected to node RL_u in Layer 3.

Layer 4 (Consequence Layer): each node in the consequence layer represents a possible consequent part of a fuzzy rule (such as "low" and "high"). The connection weights $W_{u,pk}$ of the links connecting nodes RL_u in Layer 3 to $OL_{p,k}$ in Layer 4 represent certainty factors (CFs) of the corresponding fuzzy rules when inferring fuzzy output values. Each node of this layer performs the fuzzy *OR* operation to integrate the field rules leading to the same output linguistic variables. The initial values $W_{u,pk}$ are set to unity. The output $o_{p,k}^{(4)}$ of a consequence node $OL_{p,k}$ in Layer 4 is given as follows:

$$o_{p,k}^{(4)} = \max_{u \epsilon G} (o_u^{(3)} \ W_{u,\mathrm{pk}}) \quad (4)$$

where G is the set of indices of the nodes RL_u in Layer 3 that are connected to node $OL_{p,k}$ in Layer 4.

Layer 5 (Output Layer): this layer is the defuzzification layer, where each node at this layer represents a single output variable. In this layer, either the Center of Gravity (COG) or Center of Area (COA) method can be used to compute a crisp output signal for each node. In our experiment, we used COG; the output y_p of an output node D_p in Layer 5 is given as follows:

$$y_p = \frac{\sum_{k \epsilon H} \left(o_{p,k}^{(4)} \times c_{p,k}^{(4)} \times \sigma_{p,k}^{(4)} \right)}{\sum_{k \epsilon H} \left(o_{p,k}^{(4)} \times \sigma_{p,k}^{(4)} \right)} \quad (5)$$

where H is the set of indices of the nodes $OL_{p,k}$ in Layer 4 which are connected to node D_p in Layer 5; and $c_{p,k}^{(4)}$ and $\sigma_{p,k}^{(4)}$ are respectively, the center and width of the membership function of the output linguistic value

represented by $OL_{p,k}$ in Layer 4. The weights of links from the nodes in Layer 4 to the nodes in Layer 5 are unity. Therefore, only the learnable weights in the GAFNN network are $W_{u,pk}$ between Layers 3 and 4.

3. THREE-STAGE LEARNING PROCESS OF GAFNN

3.1. Stage 1: Initialization with a Self- Organizing Algorithm

The first stage in the proposed learning approach initializes (self-organizes) the membership functions of both input and output variables of the GAFNN by determining their centers and widths. To perform this stage, we have employed a self-organizing algorithm. Alternatively, if expert knowledge is available, it can be used in this stage. Kohonen's feature-map algorithm [19] is adopted in this work to identify the initial centers $c_{n,m}^{(2)}$ and $c_{\mathrm{p},k}^{(4)}$of the membership functions which represent the input and output label nodes $IL_{n,m}$ and $OL_{p,k}$. Kohonen's algorithm is a self-organizing approach which takes a set of N-dimensional objects as inputs and produces a low-dimensional (typically two dimensional) grid, called a map.

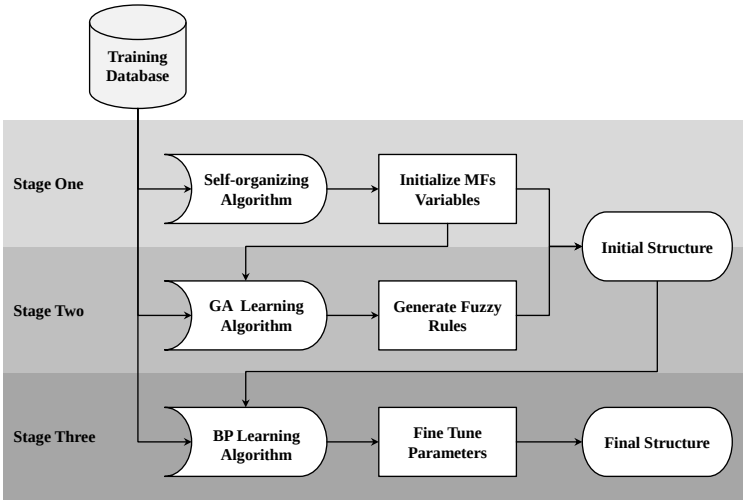


Fig. 2. Flowchart of the proposed three stage learning approach of GAFNN.

The inputs and the output of the GAFNN are represented as vectors $X^N = [x_1, x_2, \dots, x_n, \dots, x_N]$ and $Y^P = [y_1, y_2, \dots, y_p, \dots, y_P]$, where $N\ and\ P$ represent the number of the input and output variables; x_n and y_p are *n*th input and *p*th output variables respectively. It must be noted that the input and output vectors are non-fuzzy vectors. That is, each element in X^N and Y^P has a non-fuzzy value. The self-organizing algorithm adopts the following steps:

Step1: initialize $c_{n,m}^{(2)}$ *value:*

$$c_{n,m}^{(2)}(T) = \min(x_n) + \frac{1}{2}\left(\frac{m+1}{M}\right)(\max(x_n) - \min(x_n)) \qquad (6)$$

where $m \in \{1,2,\dots,M\}$, *m* is the *m*th membership function and *M* is the number of membership functions that represent the terms of the respective linguistic, T is the training iteration, and variable x_n is the *n*th element of the input vector X^N.

Step2: find the closest:

$$\left\| x_n(T) - c^{(2)}_{n,closest}(T) \right\| = \min_{1 \le m \le M} \left(\left\| x_n(T) - c^{(2)}_{n,m}(T) \right\| \right) \tag{7}$$

Step3: Update $c^{(2)}_{n,m}$ value:

$$c^{(2)}_{n,m}(T+1) = \begin{cases} c^{(2)}_{n,m}(T) + \varepsilon(T)\left[x_n(T) - c^{(2)}_{n,m}(T)\right], \\ \qquad\qquad if\ c^{(2)}_{n,m}(T) = c^{(2)}_{n,closest}(T) \\ c^{(2)}_{n,m}(T), \qquad otherwise \end{cases} \tag{8}$$

where ε(T) is a monotonically decreasing scalar learning rate.

Step4: repeat steps 2 & 3 for T=T+1, while T< the limit on time iteration.

Step5: determine the width $\sigma^{(2)}_{n,m}$ value:

$$\sigma^{(2)}_{\mathrm{n,m}} = \frac{\left| c^{(2)}_{n,m} - c^{(2)}_{n,closest} \right|}{2} \tag{9}$$

Similarly, the initial centers $c^{(4)}_{p,k}$ and widths $\sigma^{(4)}_{p,k}$ of the membership functions representing the output-label nodes can be derived, with the exception of the output vector Y^P that is used as the training data instead of the input vector X^N. It should be noted that the values of the centers and widths obtained here are all initial values which will be fine-tuned in the final stage using back-propagation learning. Since the aim of this stage is to reflect the rough locations of the clusters that are formed by the input and output data samples, any other appropriate clustering algorithm (e.g. K-means) can be used in this stage.

3.2. Stage 2: Rule Identification with GA-Based Method

The proposed GA-based method is performed in the second stage to identify the fuzzy rules that are supported by a set of training data. A simple example of GAFNN with two input linguistic variables x_1 and x_2 and one output linguistic variable *y* is considered here to explain the design process of the presented GA-based learning approach. The self-organization learning algorithm in the first stage assigns each linguistic variable a number of fuzzy sets. Let us assume that we have three fuzzy sets {low (L), medium (M), high (H)}. Then the proposed GA-based method considers all possible rules for given fuzzy sets, as shown in the top part of Fig. 3. In this simple example, there will be a total of twenty seven possible rules. In fact these rules are made up of nine possible antecedents (preconditions) of fuzzy rules represented by nodes *RL1* ...

RL9 in the Fuzzy-Rules Layer. Each antecedent has links with three possible decision fuzzy sets (nodes in Consequence Layer: L, M and H). For example, the three possible fuzzy rules associated with node *RL1* are:

If x1 is L and x2 is L, then y is L.

If x1 is L and x2 is L, then y is M.

If x1 is L and x2 is L, then y is H.

In this way the total number of rules includes all possible fuzzy rules associated with all nodes. However, at most, only one of these three fuzzy rules can be used for making decisions. We use a GA-based learning approach to identify only the appropriate and relevant fuzzy rules by filtering out all other redundant rules. A number of decisions must be made in order to implement the GA for generating appropriate fuzzy rules.

Encoding: Here we propose integer strings as chromosomes to represent candidate solutions of the problem. The string is given by $(g_1, g_2, \ldots, g_u, \ldots, g_U)$, where g_u is an integer ($0 \leq g_u \leq K$) which indicates the link of the fuzzy nodes RL_u (i.e. nodes in the Fuzzy-Rules Layer) with the output nodes (i.e. nodes in the Consequence Layer); *U* is the number of nodes in the Fuzzy-Rules Layer; and *K* is the number of neurons in the Consequence Layer. In our example shown in Fig. 3, the chromosome Ch_i has nine integers representing g_u, and $0 \leq g_u \leq 3$. The situation with $g_u = 0$ indicates there is no link between RL_u and nodes in Consequence Layer; $g_u = 1$ indicates that there is a link with '*L*' node in Consequence Layer and so on.

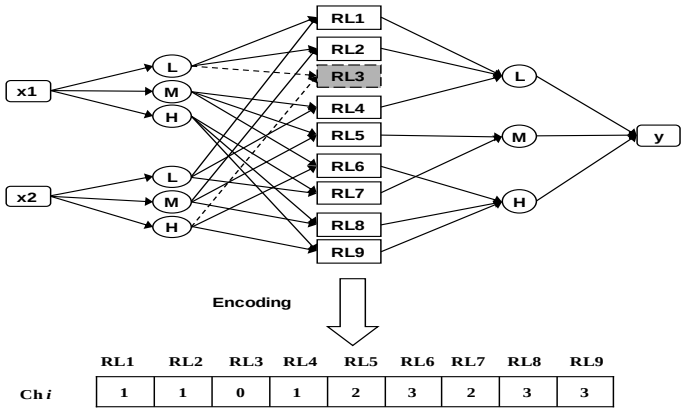


Fig. 3: An example of encoding a chromosome for possible rules with two inputs and one output

Fitness function: The GA needs a fitness value assigned to each chromosome. In this paper, we use a set of training data to calculate the fitness of each chromosome based on the following error function:

$$FIT = 1 - \left(\frac{1}{n_d} \sum_{i=1}^{n_d} (y_i - \hat{y}_i)^2 \right) \qquad (10)$$

where n_d is the number of data, y_i is the *i*th actual output, and $\hat{y}_i$ is the *i*th model output. The second component of equation (10) represents the sum of Mean Squares Errors (MSE) between actual outputs and model outputs. The GA aims to maximize this fitness function (10) in order to minimize the error value. This error value is dependent on the selected fuzzy rules.

GA operators: Based on a number of experiments, we have selected GA operators and their parameters to be used for this application. The results of those experiments are given in Section 4.2. The GA operators used are the tournament selection, the elitist generation replacement, standard two-point crossover and a random mutation [20]. The mutation operator changes the integer at each position in the solution Ch_i within the allowed range (i.e. $0 \leq g_u \leq K$) with a defined mutation probability. The initial population is created randomly. The stopping criterion for a GA run is to achieve the pre-specified error level.

When the GA learning process is completed (e.g. when a pre-specified error level is achieved), we choose the best GA chromosome. This best chromosome is decoded to get the structure of the GAFNN by keeping only the rules that are indicated by the chromosome. A gene in a GA string with $g_u \neq 0$ represents a fuzzy rule to be considered and with $g_u = 0$ to be ignored. The weight for all rules is assumed to be 1 at this stage. Then the error level (*e*) can be improved by using the back-propagation learning algorithm (stage three) to fine tune the rules weights. By doing so, we train the GAFNN with the relevant fuzzy rules only.

3.3. Stage 3: Fine Tuning Stage with a Back-Propagation Learning Algorithm

After identifying the relevant fuzzy rules and the initial structure of the GAFNN, it can adjust its parameters using the back-propagation algorithm. The aim of this learning stage is to minimize the following error function:

$$E = \frac{1}{2}(y_i - \hat{y}_i)^2 \qquad (11)$$

where y_i is the actual output and $\hat{y}_i$ is the model output for *i*th data.

From the structure of the GAFNN shown in Fig. 1, it can be observed that there are only five types of adjustable parameters. These are: centers $c_{n,m}^{(2)}$ and widths $\sigma_{n,m}^{(2)}$ of input-label membership functions, and centers $c_{p,k}^{(4)}$, widths $\sigma_{p,k}^{(4)}$ of output-label membership functions and the connection weights $W_{u,pk}$ of the links connecting nodes RL_u in Layer 3 to $OL_{p,k}$ in Layer 4 (i.e. fuzzy rules weights).

Once an input training vector X^N is presented at the input layer during supervised learning, it is propagated forward through the neural network. Subsequently, error signal $e_p^{(5)}$ is calculated in Layer 5 and then feedback to previous Layers step by step.

$$e_p^{(5)} = -\frac{\partial E}{\partial o_p^{(5)}} = o_p - o_p^{(5)} \qquad (12)$$

where o_p and $o_p^{(5)}$ are the target and actual outputs of node *p* in Layer 5.

The error signal is used to adjust the parameters. The adjustments for the centers $\Delta c_{p,k}^{(4)}$ and widths $\Delta\sigma_{p,k}^{(4)}$ of the output labels are calculated as follows:

$$\Delta c_{p,k}^{(4)} = \frac{\partial E}{\partial c_{p,k}^{(4)}} = \frac{\partial E}{\partial o_p^{(5)}} \frac{\partial o_p^{(5)}}{\partial c_{p,k}^{(4)}} = -e_p^{(5)} \frac{\sigma_{p,k}^{(4)} \; o_{p,k}^{(4)}}{\sum_k (\sigma_{p,k}^{(4)} \; o_{p,k}^{(4)})} \tag{13}$$

$$\begin{aligned} \Delta\sigma_{p,k}^{(4)} &= \frac{\partial E}{\partial \sigma_{p,k}^{(4)}} = \frac{\partial E}{\partial o_p^{(5)}} \frac{\partial o_p^{(5)}}{\partial \sigma_{p,k}^{(4)}} \\ &= -e_p^{(5)} \frac{o_{p,k}^{(4)} \; c_{p,k}^{(4)} \left(\sum_k \sigma_{p,k}^{(4)} \; o_{p,k}^{(4)}\right) - \left(\sum_k \sigma_{p,k}^{(4)} \; o_{p,k}^{(4)} \; c_{p,k}^{(4)}\right)}{(\sum_k \sigma_{p,k}^{(4)} \; o_{p,k}^{(4)})^2} \end{aligned} \tag{14}$$

Similarly, the centres $c_{n,m}^{(2)}$ and widths $\sigma_{n,m}^{(2)}$ of the membership functions of input-label nodes are adjusted as follows:

$$\Delta c_{n,m}^{(2)} = \frac{\partial E}{\partial c_{n,m}^{(2)}} = \frac{\partial E}{\partial o_{n,m}^{(2)}} \frac{\partial o_{n,m}^{(2)}}{\partial c_{n,m}^{(2)}} = \frac{\partial E}{\partial o_{n,m}^{(2)}} \; o_{n,m}^{(2)} \frac{2\left(o_n^{(1)} - c_{n,m}^{(2)}\right)}{(\sigma_{n,m}^{(2)})^2} \tag{15}$$

$$\Delta\sigma_{n,m}^{(2)} = \frac{\partial E}{\partial \sigma_{n,m}^{(2)}} = \frac{\partial E}{\partial o_{n,m}^{(2)}} \frac{\partial o_{n,m}^{(2)}}{\partial \sigma_{n,m}^{(2)}} = \frac{\partial E}{\partial o_{n,m}^{(2)}} \; o_{n,m}^{(2)} \frac{2(o_n^{(1)} - c_{n,m}^{(2)})^2}{(\sigma_{n,m}^{(2)})^3} \tag{16}$$

where

$$\frac{\partial E}{\partial o_{n,m}^{(2)}} = \begin{cases} \sum_u e_u^{(3)} & if \;\; o_{n,m}^{(2)} = \min_{i \epsilon U} \left(o_i^{(2)}\right) \\ 0 & otherwise \end{cases} \tag{17}$$

where U is the set of indices of the nodes in Layer 2 that are connected to node *u* in Layer 3.

The adjustment for the connection weights $W_{u,pk}$ is calculated as follows:

$$\Delta W_{u,pk} = \frac{\partial E}{\partial W_{u,pk}} = \frac{\partial E}{o_{p,k}^{(4)}} \frac{o_{p,k}^{(4)}}{\partial W_{u,pk}} = -e_{p,k}^{(4)} \; o_u^{(3)} \tag{18}$$

4. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we validate the performance of the GAFNN by using a well-known benchmark dataset (i.e. the Box–Jenkins time series [21]) and show the merits and capabilities of the GAFNN as compared to other models. Well-known benchmark datasets are used for the sake of easy comparison with 19 existing models published in the literature.

4.1. Nonlinear System Identification Example

Here, the GAFNN is applied to a non-linear system identification, using the gas furnace data (series J) of Box and Jenkins [21]. The data set used in this experiment was recorded from a combustion process of a methane-air mixture. During the process, the portion of methane was randomly changed, while maintaining a constant gas flow rate. The data set used here consists of 296 input-output pairs. The gas flow into the furnace is the input $x(t)$ and the CO_2 concentration in the outlet gas is the output $y(t)$. The sample interval is 9s. The characteristic function of the CO_2 time series data has significant deviation from the Gaussian behavior.

The GAFNN is employed to provide identification of the CO_2 concentration $y(t)$. In order to compare the GAFNN results with other models, a similar experimental design has been used. It is assumed that the task is to identify the CO_2 produced in a furnace $y(t)$ at time *t*, given the methane gas portion from four time steps before $x(t-4)$ and the last CO_2 produced in the furnace $y(t-1)$. The data set was converted to $[x(t-4), y(t-1): y(t)]$ pairs which reduced it to 292 input-output pairs.

All data sets used in this experiment were normalized using the *min-max* normalization technique given as follows:

$$X_n = \frac{(X - X_{\min})}{X_{max} - X_{min}} \tag{19}$$

where $X_n$is the normalized value, and X, X_{min}, and X_{max} are an instance of the minimum and the maximum values of the vector to be normalized. The normalized data was then divided into two parts with random sampling to take care of random effects. The first part 70% (204 records) was used for training the GAFNN, and the other 30% (88 records) unseen data was used for testing the trained GAFNN. Due to the stochastic nature of the approach the simulation is run 10 times, and the averaged performance index is considered (see next section).

This identification problem is then mapped onto the five-layer GAFNN with the following configuration as shown in Fig. 4. The input layer consists of two nodes: $x(t-4)$ and $(t-1)$, whereas, the output layer consists of one node: $y(t)$. The input and the output variables were divided into five linguistic labels (VS, S, M, L, and VL). Thus, the Condition Layer consists of 10 nodes and the Consequence Layer consists of 5

nodes. The initial fuzzy-rule layer consists of all possible combinations of input variables which, in this is case, is 25 nodes. They are fully connected with the Consequence Layer.

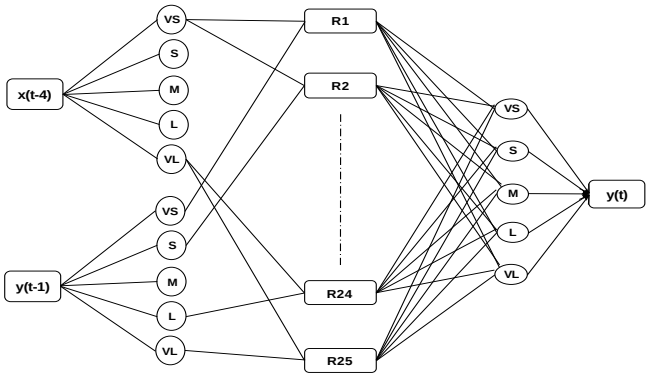


Fig. 4: The initialized GAFNN Structure for the Box-Jenkins gas furnace data.

The initial parameters (center and width) of the membership functions for all inputs and output variables were generated using the self-organizing algorithm described in Section 3.1. Fig. 5 shows these initial membership functions. While other divisions of the domain regions and other shapes of membership functions are possible, we initially divide the input and output spaces into five fuzzy regions with the same Gaussian membership functions following the example presented in [17]. In this experiment we use the Mean Square Error (*MSE*) as a performance index for the GAFNN:

$$MSE = \frac{1}{n_d} \sum_{i=1}^{n_d} (y_i - \hat{y}_i)^2 \tag{20}$$

where n_d is the number of data, y_i the actual output, and $\hat{y}_i$ is the model output for *i*th data.

4.2 GA Operations and Parameters

In order to evaluate the impact of different GA operations and parameters on the GA performance, a sensitivity analysis has been carried out. In order to avoid random effects, the experimentation approach adopted involved conducting 10 runs with a particular selection of parameters and identifying the best solution (lowest MSE) over these runs, and the average of the best solutions from each of the 10 experiments. A mutation probability of 0.05 was considered to analyze how sensitive the GA performance is for two crossover types (one point and two-point) with different crossover probabilities ranging from 0.5 to 0.9. In the same way, the sensitivity of the performance of GA to mutation probability has been analyzed using a two-point crossover with a probability of 0.7 (which was found to be the best value). Finally, using a two-point crossover with a probability of 0.7 and a mutation probability of 0.05, the sensitivity of the performance of GA to different population sizes ranging from 20 to 120 has been analyzed. The peak

performance of GA was achieved when two-point crossover with a probability of 0.7, mutation probabilities of 0.05 (and 0.06) and population sizes of 90 were used.

The stopping criterion for a GA run is to achieve the pre-specified error level (e.g. *MSE* < 0.001). The GA-based fuzzy rules identification method has been used to generate the fuzzy rules. Firstly, a candidate solution has been encoded into an integer string as a chromosome. The chromosome size has been set as 25, the total number of nodes in the fuzzy-rules layer. Each gene g_u , where $0 \leq g_u \leq 5$, represents a fuzzy rule. Then an initial population containing *N* chromosomes has been generated randomly.

4.3 Experimental Results

This second stage started with 25 fuzzy rules (i.e. all possible rules), and after completion, the number of rules was decreased to 18, with a mean square error (*MSE*) = 0.00138. The fuzzy rules generated from this stage are shown in Table 1. It should be noted that the weight of all generated rules in this stage was set to 1. Lastly, after identifying the GAFNN structure, the BP algorithm explained in Section 3.3 was employed (with learning rate φ = 0.1) to fine tune the membership parameters and the weights of the fuzzy rules. In this stage two experiments were run. In the first experiment, the BP algorithm was only employed to adjust the memberships parameters (centers, widths) as is the case in most similar methods such as [22] (i.e. the error for nodes in Layer 4 was computed and propagated without updating to $W_{u,pk}$). In the second experiment both membership parameters and fuzzy rules weights $W_{u,pk}$ were adjusted. The results of both experiments after 150 epochs are illustrated in Fig. 6. The improvement in training error rate (*MSE*) achieved in experiment one is denoted by the dotted curve, from 0.00138 to 0.00082, while the solid curve represents the improvement in error rate (*MSE)* achieved in experiment two, from 0.00138 to 0.00042.

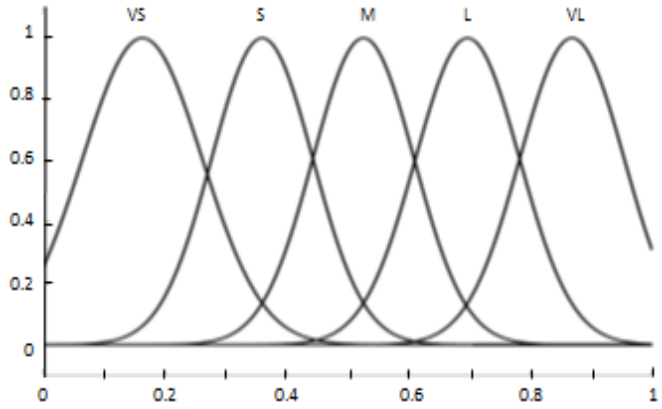


Fig. 5. The initial membership functions generated from the first stage of the learning process of GAFNN for initial input 1 $\boldsymbol{x(t-4)}$; initial input 2 $\boldsymbol{y(t-1)}$; and initial output 1: $\boldsymbol{y(t)}$.

Fig. 7(a), (b) and (c) show the learned input-output membership functions after this stage for the Box-Jenkins gas furnace data sets. The shapes, in particular the widths, of the different linguistic variables of these membership functions have evolved from their initial shapes (Fig. 5) to the shapes shown in Fig. 7

with some overlaps between them. For example, in Figs. 7(b), (c), the width of the membership function of linguistic variable "VS" is much larger than that of "S". From a linguistic variable perspective, the disorder of linguistic variables induced by learning is difficult to understand. The reasons for this could be the lack of the variety of linguistic variables, lack of the number of rules considered, or different order of the linguistic terms used in Fig 7. The fuzzy rules and their corresponding weights resulted from this stage are given in Table 1. There are three rules with zero weights, outlined by the (*) in the table, which means that those rules can be deleted without affecting the outputs. The total number of appropriate and relevant rules in this case has reduced to15 from the 25 possible fuzzy rules.

TABLE 1
FUZZY RULES GENERATED FROM THE SECOND AND THE THIRD STAGES OF THE LEARNING PROCESS AND THEIR CORRESPONDING WEIGHTS OF THE BOX-JENKINS GAS FURNACE DATA.

Rule #	IF		THEN	Weight	
	$x(t-4)$	$y(t-1)$	$y(t)$	After Stage2	After Stage3
1	VS	VS	VL	1	0.93
2	VS	S	VS	1	0.94
3	VS	M	L	1	1.00
4	VS	VL	VL	1	0.99
5	S	S	S	1	0.48
6	S	M	L	1	0.17
7	S	L	L	1	0.63
8	S	VL	VL	1	0.31
9*	M	S	S	1	0.00
10	M	M	M	1	1.00
11	M	L	M	1	0.21
12*	L	S	S	1	0.00
13	L	L	L	1	0.79
14	L	VL	VL	1	0.48
15	VL	VS	VS	1	0.68
16*	VL	M	S	1	0.00
17	VL	L	S	1	0.20
18	VL	VL	VS	1	0.20

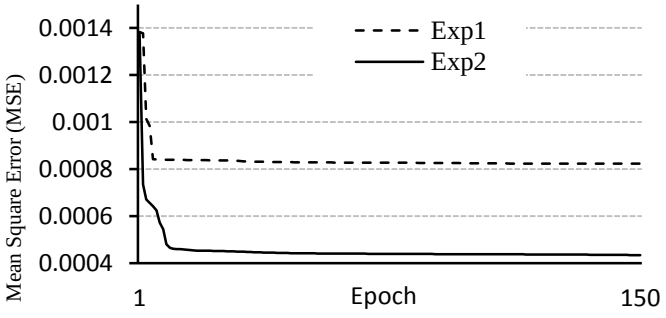
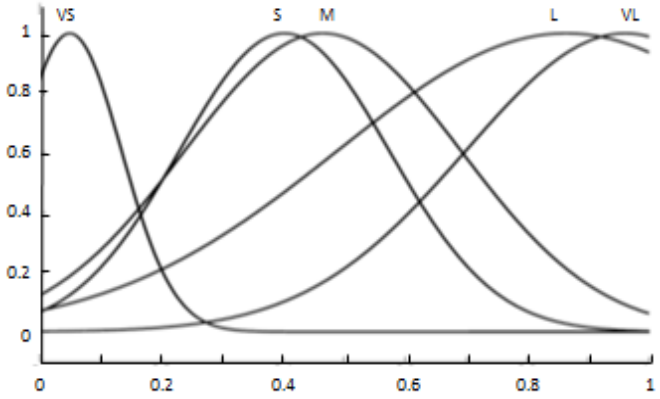
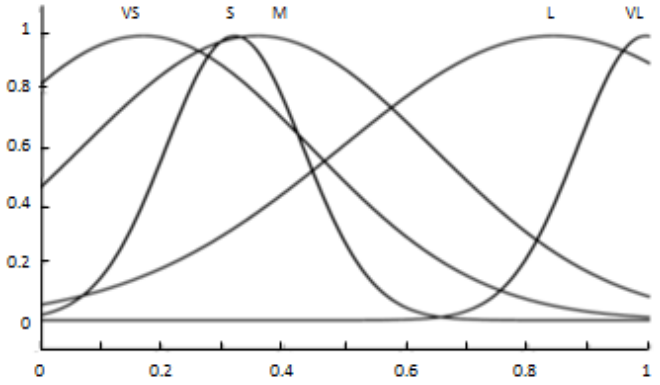


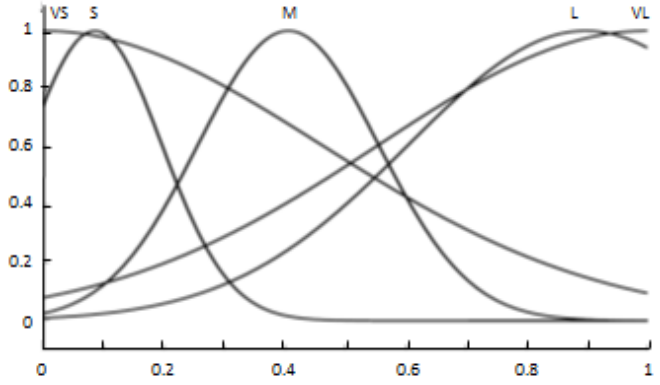
Fig. 6. The improvement of the GAFNN prediction error of the Box-Jenkins data after the BP algorithm.



(a) Final input 1: $x(t-4)$



(b) Final input 2: $\boldsymbol{y(t-1)}$



(c) Final output 1: $y(t)$

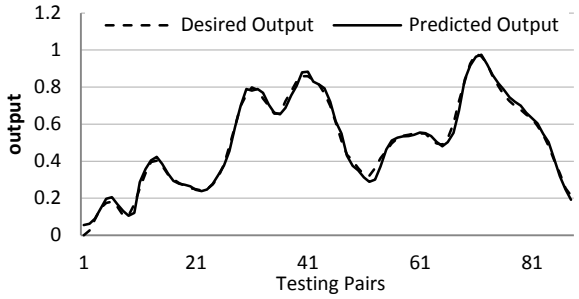
Fig. 7: The learned input-output membership functions of the Box-Jenkins gas furnace data.

After completing the training process for the GAFNN, the unseen testing data set was used to test the trained GAFNN. The GAFNN successfully identified the desired nonlinear system dynamics with a very low *MSE* (= 0.00045). Fig. 8 summarizes the final results of this experiment, where the real (desired) and the

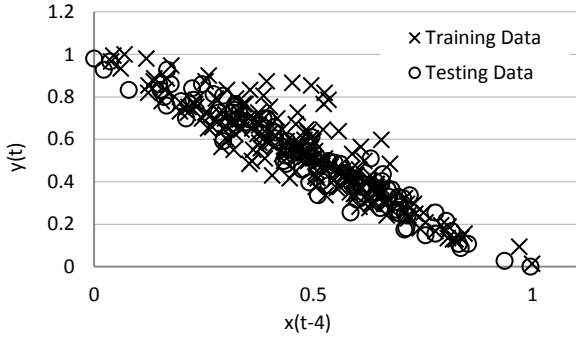
predicted outputs in response to the testing input data are given in graphical form in Fig. 8(a), and training and testing data distribution is shown in Fig. 8(b). The data distribution shows that the testing data are clustered tightly than that of the training data. Fig. 8(c) shows the GAFNN prediction error, $y_i - \hat{y}_i$, where y_i the actual output, and $\hat{y}_i$ is the model output for *i*th testing data. It can be observed from the figure that the predicted output closely fits with the desired output. The errors are within the range of [-4%, +5%].

4.4 Comparison with Other Models

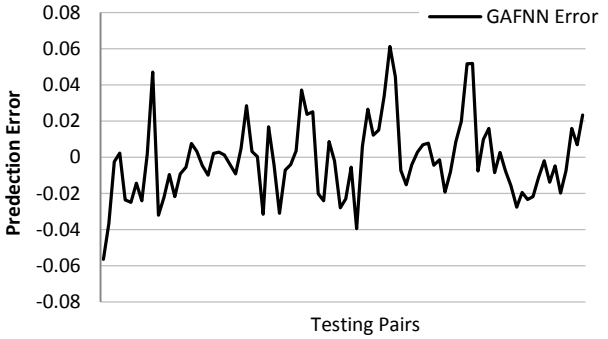
Table 2 shows a comparison of the results obtained from our proposed GAFNN and that of different models reported in the literature for the Box-Jenkins data prediction problem. It is worth mentioning that the experimental design adapted for the GAFNN in this work is similar to the other models listed in Table 2. The table summarizes the general structure of each model, including the number of inputs and the number of rules as presented in [17]. Fuzzy based neural models (such as ANFIS, FuNN, HyFIS, and GAFNN) or improved neural network model (such as Improved Neuro-Endocrine) have a much better performance with respect to their counterpart mathematical models (such as ARMA) and fuzzy models (such as Xu, Sugeno and Takagi).



(a) Desired and predicted outputs**.**



(b) Training and testing distributions.



(c) The GAFNN prediction error.

Fig. 8: The GAFNN performance of the Box-Jenkins gas furnace data.

It can be observed that the performance of our proposed GAFNN exceeds that of almost every other model, the exceptions being the HyFIS and 'Improved Neuro-Endocrine' models. It is worth noting that the HyFIS approach uses all data samples to construct the model (i.e. creating fuzzy membership functions of the input-output variables and generating the fuzzy rules) and 92 data pairs of the same data set were used for testing the model. In our case, the data set was divided into two parts. The first part was used to construct and learn the GAFNN model, then the model was tested with the second part which is unseen data. The model error presented in Table 2 for the GAFNN is for this unseen test data set. The improved Neuro-Endocrine model proposed in [37] uses the interaction mechanism of different glands for regulating the neural network. The model improved the predictive accuracy of time series. It has the smallest MSE, but the number of inputs is 10, which is much larger than other models. The paper itself rightly recognizes that the larger number of inputs might increase the computation cost of training.

Also, the comparative results show that DE–GMDH-type network [35] and FuNNs [36] are very promising models. The DE–GMDH-type network is a hybrid of the group method of data handling (GMDH) and differential evolution (DE), which is shown to outperform all models, except the last four in Table 2. The 'DE–GMDH-type network' uses 6 inputs for the gas furnace process data which is relatively higher than most other models. The FuNNs model has an acceptable error level (0.00051) with a lower number of rule nodes (7) than our GAFNN (15). FuNNs uses a multilayer perceptron (MLP) network and a modified back-propagation training algorithm. The general FuNN architecture consists of five layers: input, condition, rule, action and output layers. The second and third (condition and rule) layers are fully connected and the sigmoidal logistic activation function is used in each rule node to represent the degree to which input data match the antecedent component of an associated fuzzy rule. That means each rule node in the FuNN is represented by several fuzzy rules, each of them representing a combination of the input condition elements which would activate that node. Therefore, the number of rule nodes in FuNN, from the Gas-Furnace Time Series example 7, does not represent the total number of fuzzy rules in our GAFNN model. In general, the results of this experiment show that the proposed GAFNN is a competitive model when compared with other promising models published in the literature.

Table 2: Comparative results for different modeling approaches.

Model name and reference	Number of inputs	Number of rules	Model error (MSE)
ARMA model [21]	5	-	0.71
Tong's model [23]	2	19	0.469
Pedrycz's model [24]	2	81	0.320
Xu's model [25]	2	25	0.328
Sugeno's model [26]	2	6	0.355
Surmann's model [27]	2	25	0.160
Linear model [28]	5	-	0.193
Takagi-Sugeno model [28]	6	2	0.068
Position-gradient model [28]	3	6	0.190
Lee's model [29]	2	25	0.407
Hauptmann's model [30]	2	10	0.134
Lin's model [31]	5	4	0.261
Nie's model [32]	4	45	0.169
Pedrycz et al.'s model [33]	2	25	0.3950
ANFIS model [34]	2	25	0.00073
DE–GMDH-type network [35]	6	-	0.00053
FuNN model [36]	2	7	0.00051
HyFIS model [17]	2	15	0.00042
Improved Neuro-Endocrine [37]	10	-	0.00016
GA-FNN (current model)	*2*	*15*	*0.00045*

5. ROAD TRAFFIC CASE STUDY

This section demonstrates the application of the proposed GAFNN to a real-world road traffic prediction problem. In this experiment GAFNN is employed as a prediction tool which can be used in a road traffic control center to predict the total travel time and total distance traveled for different traffic control actions. The inputs of GAFNN are the proposed traffic control action and the current traffic state, while the outputs are the predicted total travel time (*TTT*) and total distance traveled (*TDT*).

Total travel time, which is the travel time of all vehicles in a network during a considered period, and total travel distance, which is the sum of the distances travelled by all vehicles in a network during a considered period, are two of several performance criteria used to assess the performance of a traffic control action. A traffic control action (such as metering ramps, controlling speed limits by Variable Signs (VS), using Dynamic Route Information Panels (DRIP), and/or lane closure) is a response of the road traffic control center to manage the road traffic state. So, when road traffic congestion happens, the most appropriate traffic control action should be selected in real time to manage the current traffic state. This process needs to predict of the performance of all possible control actions on the current traffic state to find the appropriate one.

5.1 Case Study Design

The traffic case study is created for a sub-network of the road traffic network in the Riyadh city of Saudi Arabia, see Fig. 9. The sub-network chosen is one of the busiest parts of the Riyadh network, as it is used mostly by traffic approaching the city center. This sub-network includes 10-km of King Fahad highway with

four lanes each way. Two main roads (*R1* and *R2*) separate from King Fahad highway and run parallel to it and then join it again. Traffic enters the section from two origins (*O1* and *O2*) and leaves it through two destinations (*D1* and *D2*). In this case study, we only consider traffic going from south to north (i.e. towards the city center). A detailed analysis of this case study can be found in [38].

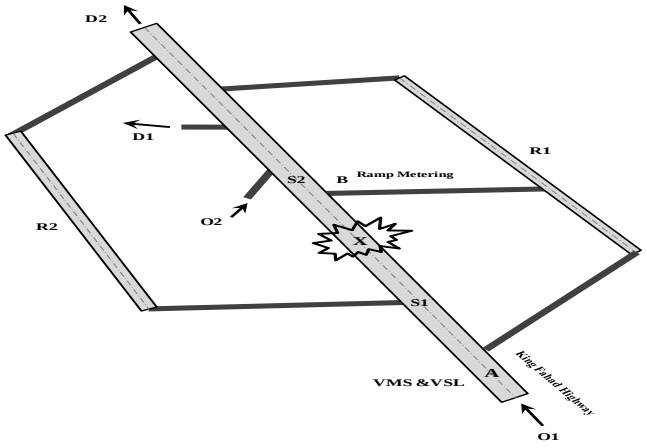


Fig. 9: The sub-network considered in the traffic case study.

We have considered the following variables in our case study:

Three traffic factors to represent the current traffic state:

- Average traffic demand (*TDm*): the average traffic inflows at the origin links of the sub-network (*O1* & *O2*).
- Average traffic density (*TDn*): the average number of vehicles per km per lane on the King Fahad highway.
- Incidents status (*IS*): only the severity of incidents is considered in this case study.

Five traffic control actions:

- ca_1: Using VMS at point *A* to direct traffic that goes to *D2* to use road *R1*.
- ca_2: Using VMS at point *A* to direct traffic that goes to *D1* to use road *R1*.
- ca_3: Using VMS at point *A* to direct traffic that goes to *D1* and traffic that goes to *D2* to use road R1.
- ca_4: Using VMS at point *A* to direct traffic that goes to *D2* to use road *R2*.
- ca_5: Using VMS at point A to direct traffic that goes to *D1* to use road *R1* and on Ramp Metering at point *B*.

The data needed for the training and testing processes of GAFNN has been generated using a traffic macroscopic simulation model (METANET macroscopic flow model) [39]. The simulation has been run many times with different combinations of traffic states and the five control actions for a prediction time intervals set to 120 minutes. From each run one data pair sample, (ca_i, *TDm, TDn, IS;* TTT_i, TDT_i) has been abstracted. ca_i, *TDm, TDn,* and *IS* are the input variables and they represent the current traffic state and the

proposed control action. TTT_i, TDT_i are the output variables and represent the predicted total travel time and predicted total distance travelled after 120 minutes of the application of control action ca_i.

Table 3: Maximum, minimum, and average value of data set for the considered traffic case study.

	TDm	*TDn*	*IS*	*TTT*	*TDT*
MIN	500.0	5.0	2000.0	807.7	76685.6
MAX	6000.0	150.0	6000.0	58606.2	352915.9
Mean	3311.6	79.4	4094.0	23244.6	23244.6
Std Dev	1608.2	43.1	1620.9	16549.2	55863.3
CV (%)	48.6	54.3	39.6	71.2	240.3

The generated data set consists of 5000 input-output pairs. Table 3 summarizes the statistical indicators for the input-output parameters, including the min, max, mean, standard deviation, and coefficient of variance. It is necessary to ensure that the generated data is sufficient to cover the entire problem domain, including maximum and minimum values for each variable, as well as a good distribution of values within this range. Using the values in Table 3, the normalization of data samples was performed using Equation (19). The normalized data samples are then divided such that 70% was used for training the GAFNN and 30% unseen data was used for testing the trained GAFNN.

5.2 Learning GAFNN

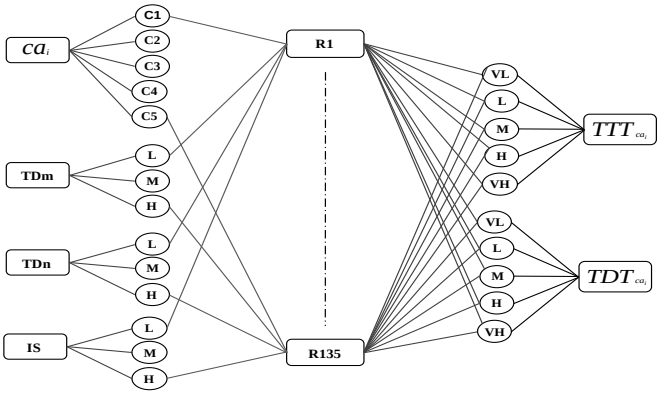


Fig. 10: The initial GAFNN Structure for the traffic case study.

The membership functions of both input and output variables were initially determined by using some expert knowledge as follows. The input variables *TDm*, *TDn* and *IS* were divided into three labels (low, medium, and high), whereas the control action ca_i was directly represented by five non-fuzzy indices (C1, C2, C3, C4, and C5). On the other hand, the output variables TTT_i and TDT_i were divided into five labels (very low, low, medium, high, and very high). Consequently, the condition layer consisted of 14 nodes and the consequence layer consisted of 10 nodes. The fuzzy-rule layer consisted of all possible combinations of

input variables which, in this case, was 135 nodes. They are fully connected with the Consequence Layer. Fig. 10 shows the overall structure of the GAFNN after this stage. The initial parameters (center and width) of the membership functions for all input and output variables are given in Table 4 in columns 3 and 4.

We started the second stage of the learning process with 135 fuzzy rules, and after completing this stage, 126 relevant fuzzy rules were identified with a mean square error (*MSE*) of 0.013. After a series of experiments the following learning parameters were identified as appropriate: (*i*) population size = 200, (*ii*) tournament selection, (*iii*) elitist generation replacement, (*iv*) standard two-point crossover with probability = 0.8, (*v*) and mutation probability = 0.04. The stopping criterion was 1,500 generations. Fig. 11 shows the GAFNN prediction error of the traffic case study after the second stage of the learning process. The curve represents an average performance of 10 experiments.

Table 4: Initial and final parameters of the MFs for all variables of the traffic case study.

I/O variables	Label	Initial Parameters		Final Parameters	
		Center	Width	Center	Width
ca_i	C1	0	0.105	0	0.105
	C2	0.25	0.105	0.25	0.105
	C3	0.5	0.105	0.5	0.105
	C4	0.75	0.105	0.75	0.105
	C5	1	0.105	1	0.105
TDm	Low	0.16	0.17	0.29	0.48
	Medium	0.50	0.17	0.51	0.41
	High	0.85	0.17	0.88	0.28
TDn	Low	0.15	0.17	0.16	0.19
	Medium	0.50	0.17	0.52	0.17
	High	0.86	0.18	0.79	0.22
IS	Low	0.2	0.21	0.14	0.40
	Medium	0.5	0.21	0.53	0.41
	High	0.9	0.21	0.98	0.32
TTT	Very	0.0592	0.09395	0.01	0.1
	Low	0.2471	0.09395	0.03	0.115
	Medium	0.4732	0.0986	0.12	0.12
	High	0.6704	0.0876	0.71	0.215
	Very	0.8456	0.0876	0.89	0.185
TDT	Very	0.2136	0.1234	0.13	0.085
	Low	0.4604	0.0871	0.14	0.035
	Medium	0.6346	0.0654	0.8	0.11
	High	0.7654	0.0654	0.85	0.115
	Very	0.9094	0.072	0.95	0.095

Due to the large number of experiments and the long training time required, a modified master-slave parallel GA (used in [40]) was used with a network of 35 PCs to speed up this stage of training. For example, the completion of one generation using only one PC takes approximately 450 seconds, which is decreased sharply to approximately 19 seconds with 35 PCs.

The BP algorithm (with a learning rate φ of 0.1) was then used to fine-tune the membership parameters and weights of identified fuzzy rules. Error rate *MSE* was improved to 0.0024 and the number of fuzzy rules decreased to 112 rules from potential maximum of 135. As can be seen from these results and experiments for the gas furnace benchmark example (section 4.3) the actual reduction in the number of rules depends on the dataset. The improvement seen in *MSE* during this stage is illustrated in Fig. 12. The final parameters (center and width) of the membership functions for all input and output variables are summarized in Table 4 (columns 5 and 6). Note that the values of the initial and final parameters for ca_i are the same, this is because ca_i are the control actions represented by non-fuzzy values.

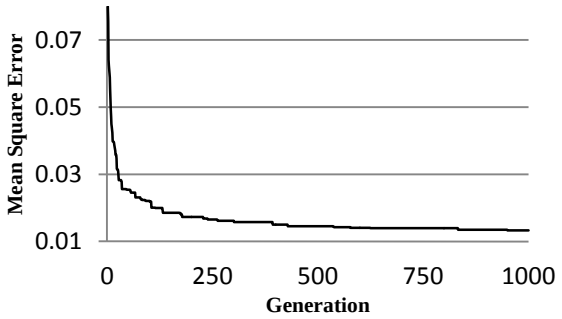


Fig. 11: The GAFNN prediction error of the traffic case study after the second stages of the learning process.

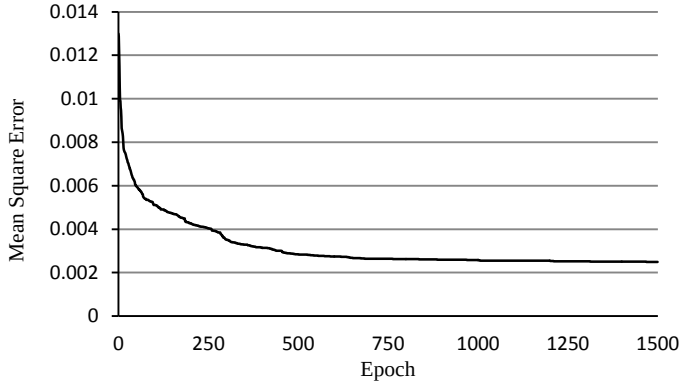
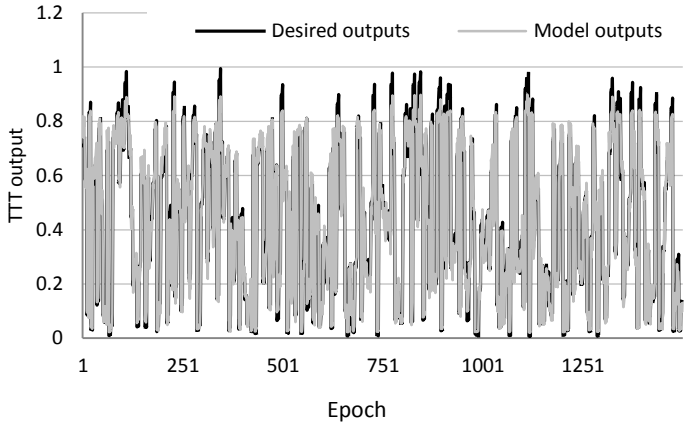
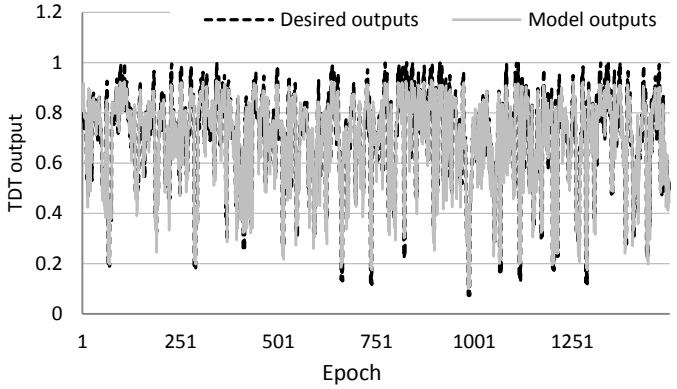


Fig. 12: The improvement of the error rate after the BP algorithm.

Finally, the trained GAFNN was employed to predict the total travel time and total distance travelled for the five control actions using the testing data. The general testing results are given in Fig. 13 and 14. The real and modeled outputs in response to the testing input data are illustrated in Fig. 13 (a) and (b), while the GAFNN prediction errors of normalized data samples are illustrated in Fig. 14 (a) and (b). We can see clearly from these figures that the GAFNN can predict total travel time and total distance travelled with a good level of accuracy.

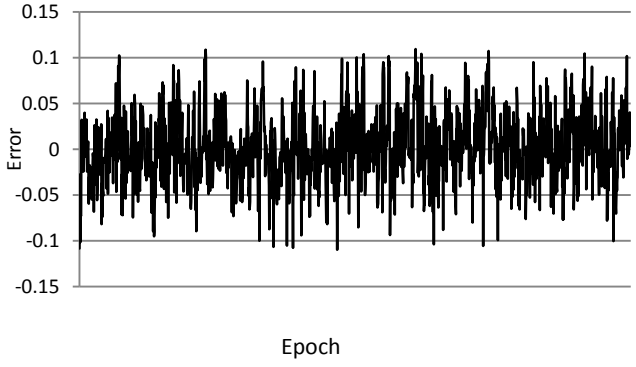


(a)

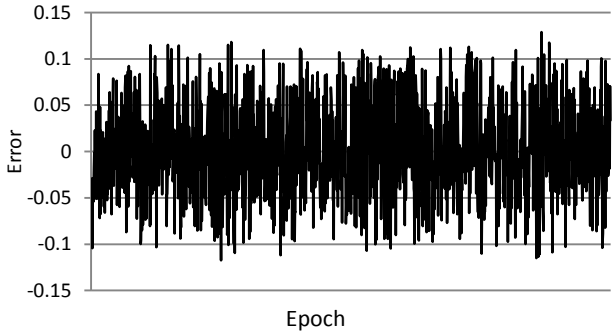


(b)

Fig. 13: (a) Desired and predicted output *TTT*; (b) Desired and predicted output *TDT*.



(a)



(b)

Fig. 14: (a) The GAFNN prediction error of *TTT*; (b) the GAFNN prediction error of *TDT*.

5.3 Result Validation

In order to illustrate how accurately the predicted data set replicates the desired output data set, we used the coefficient of determination (R^2) as a performance index of the GAFNN with $0 \leq R^2 \leq 1$. When R^2 equals zero, the desired and predicted outputs are totally uncorrelated. In contrast, when R^2 equals 1, they are exactly the same. R^2 is defined as:

$$R^2 = \frac{\left[\sum_{i=1}^{n^d}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})\right]^2}{\sum_{i=1}^{n^d}(y_i - \bar{y})^2 \ \sum_{i=1}^{n^d}(\hat{y}_i - \bar{\hat{y}})^2} \tag{20}$$

where n^d is the number of data, y_i is the *i*th actual output, $\hat{y}_i$ is the *i*th model or predicted output, $\bar{y}$ is the mean of actual output y_i , and $\bar{\hat{y}}$ is the mean of model output $\hat{y}_i$.

The results of the GAFNN performance prediction for total travel time and total distance travelled were recorded for each control action separately, and R^2 values were calculated accordingly. Table 5 summarizes the final results of this part of the experiment.

Table 5: $\boldsymbol{R^2}$ of the TTT and TDT for the five control actions.

Traffic Control Action	R^2	
	TTT	*TDT*
ca_1	0.97	0.93
ca_2	0.98	0.95
ca_3	0.98	0.95
ca_4	0.97	0.94
ca_5	0.98	0.96

Table 5 shows that the R^2 values calculated for *TTT* prediction are higher than those calculated for the *TDT* prediction. This indicates that the level of accuracy of TDT prediction is lower than that of *TTT*. Also, we can see that the GAFNN is able to predict the performance of the control action ca_5 more accurately than other control actions. It can be observed from Table 5 that R^2 values obtained from the GAFNN for *TTT* and *TDT* are high and very close to 1 ($R^2 \geq 0.93$). The higher the value of the R^2 coefficient of determination, the better is the performance of the prediction model. . This shows that the GAFNN is able to predict total travel time and total distance travelled for all control actions with good accuracy based on the performance index of the coefficient of determination (R^2).

The proposed GA-based learning approach reduces the number of rules by filtering out redundant and inappropriate rules from the possible rule set. The number of rules reduced is from 25 to 15 for the gas furnace benchmark example (section 4.3), and from 135 to 112 for the traffic case study (section 5.2). The level of reduction in the number of rules depends on the datasets. Identification of representative and relevant fuzzy rules by filtering out redundant rules generally leads to an improved performance and computational efficiency.

6. CONCLUSIONS

This paper has proposed a GA-based learning approach for FNNs. The proposed learning approach consists of three stages: first stage is initializing the membership functions of both input and output variables by determining their centers and widths using a self-organizing algorithm; a GA based learning algorithm is performed in the second stage to identify the fuzzy rules; in the last stage, the derived structure and parameters are fine-tuned by using the back-propagation learning algorithm. The structure of GAFNN is a Mamdani inference based FNN structure with five layers. The main features of the proposed GAFNN are the learning process of identifying the fuzzy rules and the learning process of adjusting rules weights in separate stages to ensure that only the relevant rules are trained.

A well-known benchmark example was used to test the performance of the proposed GA-based learning approach. Moreover, the prediction capability of the proposed system was tested for forecasting the performance of traffic control actions on the current traffic state. Experimental results have demonstrated the ability of GAFNN to identify all the relevant fuzzy rules from the training data. Comparative analysis has shown that GAFNN has a competitive degree of prediction capability than other models.

REFERENCES

[1] C.-S. Chen, "Robust Self-Organizing Neural-Fuzzy Control with Uncertainty Observer for MIMO Nonlinear Systems," *IEEE Transactions on Fuzzy Systems*, vol.19, no.4, pp.694-706, 2011.

[2] D. Coyle, G. Prasad, T.M. McGinnity, "Faster Self-Organizing Fuzzy Neural Network Training and a Hyperparameter Analysis for a Brain–Computer Interface", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 39, no. 6, pp. 1458–1471, 2009.

[3] C. Quek and R. W. Zhou, "The POP Learning Algorithms: Reducing Work in Identifying Fuzzy Rules," *Neural Networks*, vol. 14, no. 10, pp. 1431-1445, 2001.

[4] D. Shi, M.N. Nguyen, S. Zhou and G. Yin, "Fuzzy CMAC with Incremental Bayesian Ying–Yang Learning and Dynamic Rule Construction", IEEE Transactions on *Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 40, no. 2, pp. 548 – 552, 2010.

[5] F. Castro, A. Nebot, and F. Mugica, "On the extraction of decision support rules from fuzzy predictive models", *Applied Soft Computing*, vol. 11, no. 4, pp. 3463-3475, 2011.

[6] C. Hsu, C. Lin, R. Yeh, "Supervisory adaptive dynamic RBF-based neural-fuzzy control system design for unknown nonlinear systems", *Applied Soft Computing*, vol. 13, no. 4, pp. 1620-1626, 2013.

[7] M. Chen, Q. Wang, and Y. Yang, "A Hybrid Knowledge-Based Neural-Fuzzy Network Model with Application to Alloy Property Prediction", *Lecture Notes in Computer Science*, vol.4491, pp.528-535, 2007.

[8] X. Luo, W. Hou, Y. Li, and Z. Wang, "A Fuzzy Neural Network Model for Predicting Clothing Thermal Comfort," *Computers and Mathematics with Applications*, vol. 53, no. 12, pp. 1840-1846, 2007.

[9] W. Y. Wang, C. Y. Cheng, and Y. G. Leu, "An Online GA-Based Output-Feedback Direct Adaptive Fuzzy-Neural Controller for Uncertain Nonlinear Systems," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 34, no. 1, pp. 334-345, 2004.

[10] M. Mashinchi, A. Selamat, "An improvement on genetic-based learning method for fuzzy artificial neural networks", *Applied Soft Computing*, vol. 9, no. 4, pp. 1208-1216, 2009.

[11] G. Leng, T. M. McGinnity, G. Prasad, "Design for Self-Organizing Fuzzy Neural Networks Based on Genetic Algorithms", IEEE Trans, Fuzzy Syst., vol.14, no.6, pp.755-766, 2006.

[12] F. Hoffmann, "Combining Boosting and Evolutionary Algorithms for Learning of Fuzzy Classification Rules," *Fuzzy Sets and Systems*, vol. 141, no. 1, pp. 47-58, 2004.

[13] A. González, R. Pérez, Y. Caises and E. Leyva, "An Efficient Inductive Genetic Learning Algorithm for Fuzzy Relational Rules", *International Journal of Computational Intelligence Systems*, vol. 5, no. 2, pp. 212-230, 2012.

[14] H. Ishibuchi and T. Yamamoto, "Fuzzy Rule Selection by Multi-Objective Genetic Local Search Algorithms and Rule Evaluation Measures in Data Mining," *Fuzzy Sets and Systems*, vol. 141, no. 1, pp. 59-88, 2004.

[15] K. Almejalli, K. Dahal, and A. Hossain, "GA-Based Learning Algorithms to Identify Fuzzy Rules for Fuzzy Neural Networks," in *The 7th International Conference on Intelligent Systems Design and Applications, IEEE Computer Science, ISDA2007*, Rio de Janeiro, Brazil, 2007, pp. 289-296.

[16] P. Castro and H. Camargo, "Focusing on Interpretability and Accuracy of a Genetic Fuzzy System," in *The 14th IEEE International Conference on Fuzzy Systems*, Reno, Nevada (USA), 2005, pp. 696-701.

[17] J. Kim and N. Kasabov, "HyFIS: Adaptive Neuro-Fuzzy Inference Systems and Their Application to Nonlinear Dynamical Systems," *Neural Networks*, vol. 12, no. 9, pp. 1301-1319, 1999.

[18] C. Quek, M. Pasquier, and B. B. S. Lim, "POP-TRAFFIC: A Novel Fuzzy Neural Approach to Road Traffic Analysis and Prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 2, pp. 133-146, Jun. 2006.

[19] T. Kohonen, Self-Organization and Associative Memory, 2nd ed. Berlin, Germany: Springer-Verlag, 1988.

[20] K.P. Dahal, G.M. Burt, J.R. McDonald and A. Moyes, "A case study of scheduling storage tanks using a hybrid genetic algorithm" *IEEE Transactions on Evolutionary computation*, vol.5, pp.283-294, 2001.

[21] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day, 1970.

[22] D. Middleton, D. Gopalakrishna, and M. Raman, "Advances in Traffic Data Collection and Management," Texas Transportation Institute, Cambridge Systematics, White paper BAT-02-006, 2003.

[23] R. M. Tong, "The Evaluation of Fuzzy Models Derived from Experimental Data," *Fuzzy Sets and Systems*, vol. 4, no. 1, pp. 1-12, 1980.

[24] W. Pedrycz, "An Identification Algorithm in Fuzzy Relational Systems," *Fuzzy Sets and Systems*, vol. 13, pp. 153-167, 1984.

[25] C. W. Xu and M. A. Lehr, "Fuzzy Model Identification and Self-Learning for Dynamic Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 17, no. 4, pp. 683-689, 1987.

[26] M. Sugeno and T. Yasukawa, "Linguistic Modelling Based on Numerical Data," in *IFSA '91, Brussels: Computer, Management & Systems Science.*, Brussels, 1991, pp. 264-267.

[27] H. Surmann, A. Kanstein, and K. Goser, "Self-Organizing and Genetic Algorithms for an Automatic Design of Fuzzy Control and Decision Systems," in *the First European Congress on Fuzzy and Intelligent Technologies, EUFIT' 93*, Aachen, 1993, pp. 1097-1104.

[28] M. Sugeno and T. Yasukawa, "A Fuzzy-Logic-Based Approach to Qualitative Modelling," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 1, pp. 7-31, 1993.

[29] Y. C. Lee, C. H. Hwang, and Y. P. Shih, "A Combined Approach to Fuzzy Model Identification," *IEEE Transactions on System, Man and Cybernetics*, vol. 24, no. 5, pp. 736-744, 1994.

[30] W. Hauptmann and K. Heesche, "A Neural Net Topology for Bidirectional Fuzzy-Neuro Transformation," in *the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE/IFES)*, Yokohama, 1995, pp. 1511-1518.

[31] Y. Lin and G. A. Cunningham III, "A New Approach to Fuzzy-Neural System Modelling," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 2, pp. 190-198, 1995.

[32] J. Nie, "Constructing Fuzzy Model by Self-Organising Counterpropagation Network," *IEEE Transactions on System, Man and Cybernetics*, vol. 25, no. 6, pp. 963-970, 1995.

[33] W. Pedrycz, C. F. L. Patrick, and A. F. Rocha, "Distributed Fuzzy System Modelling," *IEEE Transactions on System, Man and Cybernetics*, vol. 25, no. 5, pp. 769-780, 1995.

[34] J. S. R. Jang, S. C.T., and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, Englewood Cliffs, NJ, 1997.

[35] G.C. Onwubolu, "Design of hybrid differential evolution and group method of data handling networks for modeling and prediction", Information Scinces, Vol 178, pp. 3616-3634, 2008.

[36] N. Kasabov, J. Kim, M. Watts, and A. Gray, "FuNN/2-A Fuzzy Neural Network Architecture for Adaptive Learning and Knowledge Acquisition," *Information Sciences*, vol.101, no.3, pp.155–175, 1997.

[37] D. Chen, J. Wang, F. Zou, W. Yuan, and W. Hou, "Time series prediction with improved neuro-endocrine model", Neural Computing & Application, Vol. 24, pp. 1465–1475, 2014.

[38] K.P. Dahal, K. Almejalli and M.A. Hossain, "Decision Support for Coordinated Road Traffic Control Actions", *Decision Support Systems*, Elsevier, vol 54, no. 2, pp. 962-975, 2013.

[39] Simulation-Laboratory and A. Messmer, "METANET: A Simulation Program for Motorway Network," Technical University of Crete, Dynamic Systems, Jul. 2000.

[40] S. Remde, P. Cowling, K. Dahal, N. Colledge, E. Selensky, "An Empirical Study of Hyperheuristics for Managing Very Large Sets of Low Level Heuristics", *Journal of the Operational Research Society (*JORS), vol 63, no 3, pp 392-405, 2012.